(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 March 2004 (18.03.2004)

PCT

(10) International Publication Number
WO 2004/023430 A2

(51) International Patent Classification[7]: G09B 1/00

(21) International Application Number:
PCT/EP2003/009786

(22) International Filing Date:
3 September 2003 (03.09.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/232,810 3 September 2002 (03.09.2002) US

(71) Applicant: SAP AKTIENGESELLSCHAFT [DE/DE]; Neurottstrasse 16, 69190 Walldorf (DE).

(72) Inventors: DORNER, Elmar; Hirschstrasse 40a, 76133 Karlsruhe (DE). GROSSMANN, Markus; Schelmenweg 6, 71665 Vaihingen (DE).

(74) Agent: RUTETZKI, Andreas; Müller-Boré & Partner, Grafinger Str. 2, 61671 München (DE).

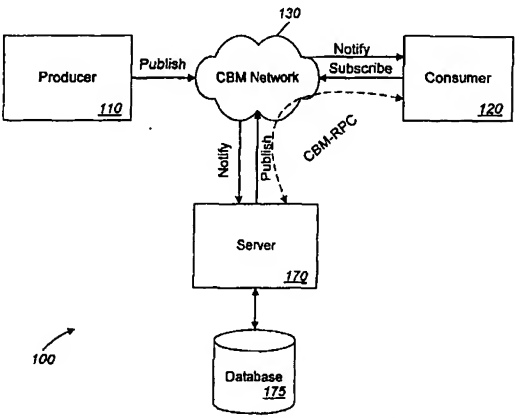
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: TUTOR INTERFACE WITH CONTENT BASED MESSAGING



(57) Abstract: A Content Based Messaging (CBM) system is provided to monitor the actions of learners, which are published as events to a CBM network. The network provides notification of the events to subscribing applications. Because CBM systems are inherently stateless, to ensure persistence of data, applications use a remote procedure call (RPC)-CBM protocol to initiate publishing to the application and retrieve historical data. After the initial CBM-RPC, messages are published in a normal CBM fashion. A tutor interface also is provided to monitor and assist students using an e-learning system. The interface is populated with information using the CBM infrastructure. The interface allows tutors to provide individualized and personal attention to each student and their learning needs.

WO 2004/023430 A2

TUTOR INTERFACE WITH CONTENT BASED MESSAGING

TECHNICAL FIELD

The following description relates generally to e-learning and in particular to content based messaging for e-learning.

BACKGROUND

Systems and applications for delivering computer-based training (CBT) have existed for many years. However, CBT systems historically have not gained wide acceptance. A problem hindering the reception of CBT systems is the lack of traditional management and oversight that is common in classroom training. Merely providing training material online does not guarantee that a learner will comprehend and learn the training material because many learners have difficulty with self-guided learning.

Although online help provides some assistance to the learner, it is difficult for an instructor or teaching assistant to gain insight about the current state and needs of an online learner absent some direct communication with the learner. Some insight may be gained from review of exercises, problems, and testing material, which can be used to gauge the learner's comprehension of the training material and the learner's progress.

In this respect, many CBT systems fail or fall short of their true potential because they do not provide for the use of dynamic information about a learner's activities. Therefore, for the above and other reasons, new methods and technology are needed to supplement traditional computer based training and instruction.

SUMMARY

In one general aspect, a tutor system includes an input to receive one or more notifications of published learner events and a processor to generate subscriptions to the notifications, to process the received notifications, and to generate a tutor interface based on the processed notification, the interface including a course area to display dynamic learner state information. The course area may display a list of learners monitored by the tutor, an indication of whether a learner is online, an indication of whether a learner is interacting with a course, a current learner action with a course, a position of a learner within a course, and a learner diary feature.

Selection of the learner diary feature causes the display of a learner diary. The learner diary may display one or more actions of a learner over a period of time. The period of time may be a week where each day is displayed as a section in the diary display.

The interface may include a watch list display including one or more courses monitored by the interface. Selection of a course from the watch list may cause the course area to populate with information based on the selected course.

The interface also may include a message area to provide a message service.

The processor may be configured to generate a CBM-remote procedure call (RPC) to access a remote function. The input may be configured to receive a return notification based on the remote procedure call causing the processor to populate the interface with information based on the return notification. The information based on the return notification may be learner state information. The input also may be configured to receive notifications based on learner update events produced by a remote service. The processor may be configured to populate the interface with information based on the learner update event notifications. The learner update information may be based on a modification to a dynamic state of one or more learners.

In another general aspect, a graphical user interface for a tutor application includes a course area to display information about a course based on a notifications of published events received from a content based messaging network, the course area displaying a list of one or more learners taking the course and information about the one or more learners on the list.

In another general aspect, a method of providing a tutoring interface includes subscribing to one or more learner events; receiving one or more notifications based on the subscribed to events; and generating a tutor interface based on received notifications, the interface including learner information. The method may include subscribing to a content based message network.

In another general aspect, a computer readable medium may include instructions for causing a processor to receive one or more notifications of published learner events; and generate subscriptions to the notifications, process the received notifications, and to generate a tutor interface based on the processed notification, the interface including a course area to display dynamic learner state information. The instructions may cause the processor to populate the course area with a list of learners monitored by the tutor.

Other features and advantages will be apparent from the description, the drawings, and the claims.

DESCRIPTION OF DRAWINGS

FIG. 1 is an exemplary block diagram of a content based messaging system including the extension of a remote procedure call.

FIG. 2 is an example of an e-learning system with content based messaging including the extension of a remote procedure call.

FIG. 3 is an exemplary process for an e-learning tutoring application.

FIG. 4 is an exemplary screen shot of an e-learning tutor user interface.

FIG. 5 is an exemplary screen shot of an e-learning tutor user interface.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

Content Based Messaging System with Remote Procedure Call Protocol

A content based messaging (CBM) system, such as, for example, Elvin, CosNotif, JMS, Keryx, and Gryphon may be used to distribute information to users, clients, and applications. As shown in Fig. 1, a CBM system 100 may include one or more information producers 110 (e.g., programs or applications running on a network device, such as a processor, workstation, or server), one or more information consumers 120 (i.e., programs or applications running on a network device, such as a processor, workstation, or server), and a CBM network 130 (e.g., one or more network devices, such as a server and associated data transportation media) for receiving and distributing information. Although Fig. 1 shows only one producer and one consumer, the CBM system 100 may include multiple producers and consumers. In addition, an application may be both a producer and a consumer of information.

An information consumer registers interest (i.e., a subscription) in specific information with the CBM network 130. An information producer 110 publishes information to the CBM network 130 corresponding to an event. An event may correspond to any action taken by a program or application. The CBM network 130 provides notifications of the published information to those consumers 120 who subscribe to the information corresponding to an event.

The CBM network 130 may include one or more data processing and distributions devices (e.g., servers, associated communications media, and data transport systems). For example, the CBM network 130 may include one or more filtering servers that receive published information and generate notifications that are transmitted to consumers 120 who subscribe to the information. The filtering server may compute the registered subscriptions that match a published event and generate a notification (i.e., a description of the real world occurrence) that is sent to the consumers 120 determined from the computed subscriptions.

The CBM system 100 provides event-driven network communications that allow essentially real-time communication of information between applications by avoiding communication delays and wasted network bandwidth associated with polling for data. In addition, processing and overhead associated with addressing may be greatly reduced because each producer and each consumer do not need to know about each other (and their addresses).

The CBM system 100 also may include a server 170 providing one or more services, procedures, and/or methods that publish information to the CBM network 130. The server 170 may have an associated database or storage device 175 to store data used by the server 170 and its services, procedures, and methods. A procedure is a function that takes one or more parameter values and returns a function value or fault. A service may provide an interface for a set of procedures. A service may function as a container for a set of procedures. A service includes a service class and the semantics for each of the procedures in the service class. The semantics include an explanation of the procedures' functionality. The functionality of procedures in a service class may be related or similar.

Typically, a CBM system does not provide a way for a consumer to access a remote procedure (e.g., a procedure running on another network device) because the consumer 120 is unaware of the location (i.e., the address) of the procedure. In order to preserve the benefits of a distributed messaging system (e.g., reduced processing and bandwidth), while providing for remote access of procedures, a content based messaging-remote procedure call (CBM-RPC) protocol is provided.

The CBM-RPC protocol is an extension of the CBM network 130 that provides a way for consumers 120 (e.g., clients/applications) to make a function or a procedure call across the CBM network 130, access a remote procedure, and receive data (e.g., a return value) generated by the procedure.

As explained above, a consumer 120 of a CBM network 130 may subscribe to an event; however, the consumer in a conventional CBM system may not request information directly from a procedure. Instead, a consumer 120 waits for notifications of published information from the CBM network 130 based on a content type of the published information. In other words, an application may register with the CBM network 130 to receive notifications of published events; however, the application does not request information across the CBM network 130 directly from a procedure.

The CBM-RPC protocol is an extension of a CBM system that allows an application to directly access a remote procedure and, for example, obtain data returned by the procedure (e.g., stored in an associated database or storage device). A consumer 120 may specify a procedure name (e.g., a unique identification (ID), such as a uniform resource identification (URI)) and parameters (e.g., a list of name/value-pairs) in a request (or a CBM-RPC), which is published to the CBM network 130. One or more, procedures or functions may subscribe to the request and generate responses (e.g., the requested information or a fault if the request is not valid) that are published to the CBM network 130.

Each procedure in the CBM system may be differentiated by an identifier. The procedure identifier may include three parts, for example, a namespace, a service class, and a procedure name. The namespace may be used to distinguish between procedures of different system applications that have the same name, and to group all procedures and service classes of one application, so that components of the CBM system may recognize the procedures associated with an application. For example, a URI or other unique identifier may be used as namespace identifier. Each application may have its own procedure namespace and its own semantics of procedures implemented within an application. For example, application 1 and application 2 may both have a procedure named "getUser"; however, the procedure of application 1 may return "Elmar" while the procedure of application 2 returns "334." Therefore, the semantics of the procedures are different (i.e., name versus user ID number).

Procedures also may be grouped into logical procedure families called "service classes." A service class may define a service and be used to instantiate the service. In the following description, a "service" may be a combination of a service class name and procedures associated with the service class. The procedure name defines a name used to identify a procedure.

Each procedure may have one or more associated parameters and a return value. Both the parameters and the return value should use data types recognized by the CBM system 100. Using Elvin CBM as an example, four different data types (e.g., "int32", "int64", "real64" and "string") are supported that may be used for names, parameters, returns, and values in a notification.

If a particular CBM protocol does not provide a composite data type (e.g., a record or an array) the following composite data type may used. The composite data type identifier may be "composite." The data type may be constructed using an XML markup style and stored in a CBM notification using an ordinary string value. For example, the structure of the markup of a course member may be implemented as follows:

```
<composite>
    <string> CourseMember1 </string>
    <int32> 32 </int32>
</composite>
```

In this example, the start delimiter and end delimiter are "composite," and the tag names for the entries may be CBM datatypes.

The CBM system 100 provides for multipoint-connections. For example, two or more services having the same name may exist in the same namespace, each of which may generate a response to a CBM-RPC request. As a result, the CBM-RPC protocol may ensure that an application (i.e., a consumer or a client) only receives responses from those procedures that were called by the application using a unique identifier.

The CBM-RPC protocol provides for directed communication that is unambiguously addressed using the unique procedure identifier. Each server having registered procedures and each client running applications are provided with a unique identifier. For example, the URI of the server implementing the called procedure and the client making the call may be used as a unique identifier by the CBM-RPC protocol. Addressing may be implemented using "from" and "to" fields, as described in further detail below. A discovery protocol may be used to support the exploration of unique identifiers and their associated procedures, as explained below.

A CBM-RPC request may include the identifiers of the client (i.e., the calling application), the namespace, the server (running the called procedure), the service class, and/or the procedure. The CBM-RPC request identifiers of the CBM-RPC request may be used to distinguish between different CBM-RPCs for the same procedure. The CBM-RPC request also may include one or more parameters used by a called procedure.

A client (e.g., an application for a tutor interface) may need to determine data (e.g., the users taking an e-learning course). A procedure running on a server may provide the name of the learners taking a course. The client publishes a request to the CBM network 130 for a procedure named "getCourseMembers" of service class "course." The procedure takes a parameter of type "string" for the course name and returns a value of composite (e.g., for the names of learners taking the course). An example of such a CBM-RPC request may be implemented as follows:

CBM.rpc.request:	1
minor:	0
from:	"http://192.168.0.0/PresenceServlet"
to:	"http://learningsolution/stateserver"
namespace:	"http://www.sap.com/cbm/elearning"
requestid:	2783462725871
serviceclass:	"course"
method:	"getCourseMembers"
params:	"<composite><string>course</string></composite>"
course:	"sapcourse20"

"CBM.rpc.request" may be used to identify the major version of the protocol that is used, and "minor" may be used to indicate a minor version of the protocol that is used. The protocol versions may be used by the CBM system 100 to ensure that a compatible or correct version of the protocol is being used. "from" may be a unique identifier (e.g., a URI) of the client making the procedure call. "to" may be a unique identifier (e.g., URI) of a server to which the procedure is registered. The request ID is a unique number identifying requests so that an application may distinguish between calls and match calls to responses. The method is the name of the method used to implement the procedure. The "params" field indicates the names of the procedure parameters. Using the parameter names, the name/values pairs of the procedure parameters may be identified. The params field also indicates the correct order of the parameter values. The "course" field is the name of the course for which the information is requested. After receiving a request notification from the CBM-network 130. The server may identify the called procedure and execute the procedure to determine a result. The procedure may return a response.

A response to a request may be similar in format to the request. For example, if a response is successfully determined (e.g., the procedure was found, executed correctly, and return a result), then the CBM-RPC response is similar to the request, except that the

"from" and "to" elements have been interchanged and the "params" element is replaced by a "returns" element (e.g., which includes the information requested by the application/client making the procedure call). A complete response to the CBM-RPC example described above may look like:

CBM.rpc.response:	1
minor:	0
from:	"http://learningsolution/stateserver"
to:	"http://192.168.0.0/PresenceServlet"
requestid:	2783462725871
namespace:	"http://www.sap.com/cbm/elearning"
serviceclass:	"course"
method:	"getCourseMembers"
returns:	"<composite><string>Markus@tutorsolution</string> <string>tiki@learningsolution</string></composite>"

If there is a problem processing the CBM-RPC request, the response notification may contain a "fault" element instead of a "returns" element. The fault element may include a string value that describes a problem encountered trying to process the request. For example, a fault indicating the requested procedure does not exist may look like:

CBM.rpc.response:	1
minor:	0
from:	"http://learningsolution/stateserver"
to:	"http://192.168.0.0/PresenceServlet"
requestid:	2783462725871
namespace:	"http://www.sap.com/cbm/elearning"
serviceclass:	"course"
method:	"getCourseMembers"
fault:	"no such procedure!"

The CBM-RPC protocol does not specify a description for procedures. Therefore, clients may not know in advance the semantics and syntax of each individual procedure. As a result, clients need to be able to determine the procedure identifier, the parameters, parameter types, the return values, and return value data types associated with a particular procedure to make a remote call of the procedure. A discovery protocol may be used to explore registered services that may be called by an application. For example, the discovery protocol may be used to determine if a service is available or unavailable.

According to the discovery protocol, the client publishes a discovery request for a service to the CBM network 130. Each service class may subscribe to the CBM network 130 for discovery requests. The CBM network 130 determines all registered services matching the request and sends a notification of the request to the service. Each service then sends a response to the requesting client. The discovery request may include a namespace identifier and/or a service class. Discovery requests that only include the namespace are sent to all registered services in the namespace. If the service class name is specified in addition to the namespace identifier, the request is set to service class having the corresponding service class name within the identified namespace are returned. The service class responds to the request.

A client may specify a service class if the client knows the name of the procedure, but not the server where the procedure is registered. For example, a discovery request to explore all procedures with the service class name "course" in the given namespace http://www.sap.com/cbm/elearning may be expressed as follows:

CBM.rpc.discovery:	1
minor:	0
from:	"http://192.168.0.0/PresenceServlet"
namespace:	"http://www.sap.com/cbm/elearning"
serviceclass:	"course"

Information responses to the discovery requests are similar to discovery requests. The only information reported in the response is the identifier of the server where the service is registered (e.g., the "from" field of the CBM-RPC). The "to" field may be included but is not necessary. For example, an information response to the service request in the example above may be:

CBM.rpc.info:	1
minor:	0
from:	"http://learningsolution/stateserver"
to:	"http://192.168.0.0/PresenceServlet"
namespace:	"http://www.sap.com/cbm/elearning"
serviceclass:	"course"

A CBM-RPC library may be added to applications using the CBM system to make procedure calls. A protocol implementation of CBM may be stored an application library.

Procedures that may be implemented in conjunction with a CBM-RPC may be implemented using code that supports the basic data types described above.

Example of Procedure and Call in a CBM system

An example of a procedure "getCourseMembers" that returns the names of members that are currently online is defined in a service class called "course" and may be expressed as follows:

```
public class Course {
private Composite Members = new Composite("John","Sam");
public Composite getCourseMembers (string course) {
Composite list  = course.Members);
return list ;
}
}
```

The getCourseMembers method of the course class uses a string value representing the name of the course.

Making the getCourseMembers method available using a CBM-RPC in this example includes two steps. First, the method must be registered with the CBM-RPC server and the server has to make a connection to the CBM network. This may be implemented as follows:

```
System.out.println("Attempting to start CBM-RPC Server...");
String serverId = "http:/learningsolution/stateserver";
CBMRpcServer server = new CBMRpcServer(CBMURL, serverId);

System.out.println("Started successfully");

// Register our handler class as "course"
ServiceIdentifier serviceId =
new ServiceIdentifier(serverId, "http://www.sap.com/cbm/elearning", "course");
Service s = new Service(serviceId, new Course());
server.addService(s);

System.out.println("Registered Course class to server.");
```

The client has to connect to the CBM network as well. The following client code fragment shown below may be used to make this connection.

```
// Connect client to CBM network
String clientId = "http://192.168.0.0/PresenceServlet";
CBMRpcClient client = new CBMRpcClient(CBMURL, clientId);

// Create the request parameters
string courseName = "BetterSale";
Parameter params = new Parameter();
params.put("course", courseName);

// Issue a request and extract result
String serverId = "http://learningsolution/stateserver";
String namespace = = "http://www.sap.com/cbm/elearning";
String serviceClass = "course";
ServiceIdentifier serviceId
= new ServiceIdentifier(serverId, namespace, serviceClass);

Object result = client.execute(serviceId, "getCourseMembers", params);


String compositeMarkup = (String) result;
Composite members = new Composite(compositeMarkup);
System.out.println("Course members logged on are"+members.toString());
```

After making the connection, the client specifies the procedure identifier and the parameter values for the procedure. With that information the procedure may be called. The call returns a result in form of an Object or it throws a fault (i.e., a Java Exception).

E-learning CBM

As shown in Fig. 2, a CBM e-learning system 200 may be used to implement CBM for e-learning applications. The CBM system 200 may include one or more applications that produce or publish information, for example, a learner application 210, and one or more applications that subscribe to the published information, for example, a tutor application 220. The learning applications may be applications used by learners in

an e-learning system. For example, the learner applications 210 may provide course material to the learner to take a course (e.g., including information of various multi-media types, such as text, audio, video, and graphics, exercises, tests, and collaborative communications, such as, a chat session, e-mail, or instant messaging) or the application may provide administrative support for the learning system (e.g., booking courses, managing a curriculum, and searching for information). The applications 210, 220 may provided by one or more servers of the CBM learning system 200 and/or by a learning station/client (e.g., that allows the learner to connect to the network and run applications).

The CBM network 230 may include one or more data processing and distributions devices (e.g., servers, associated communications media, and data transport systems). For example, the CBM network 230 may include one or more filtering servers that receive published information and generate notifications that are transmitted to applications who subscribe to the information. The filtering server may compute the registered subscriptions that match a published event and generate a notification (i.e., a description of the real world occurrence) that is sent to the applications determined by from the computed subscriptions.

During a training session, a learner interacts with an e-learning system using the learner applications 210. Throughout each learning session, a learner continually performs various actions as the learner interacts with the applications. For example, a learner may login, log out, join a course, leave a course, start a course, finish a course, navigate within a course, complete an exercise, read or post messages to newsgroups, join a chat session, leave a chat session, and send a chat message. Each of these exemplary actions may be regarded as an "event." Each event may be published to the CBM network 230 to notify other applications that subscribe to the learner events. The subscribing applications may be any programs running on one or more servers or clients that request notifications for events in the e-learning system (e.g., a tutor application to monitor and manage one or more learners assigned to a tutor).

As a learner performs various actions within a course causing events to take place, the state of the learner changes (e.g., the learner's position within a course). However, a CBM network is stateless system. In other words, once an event is published, the CBM network 230 does not save notifications of past published events. As a result, applications logging one the CBM network 230 after events have been published are not automatically notified of these events. Therefore, to ensure persistence of published learner events, a state system 240 may be provided. The state system 240 may include

one or more servers. The servers may provide one or more state services 245 that subscribe to one or more learner events.

When a learner event is published to the CBM network 230, the CBM network computes any subscriptions matching the event. The CBM network generates a notification of the event, which is sent by the CBM network to the subscribing state service (in addition to any applications subscribing to the event). The state service may process and store information based on the notification in one or more a state databases 248 and/or other storage devices. The state service 245 also may publish state update events to the CBM network when there is a modification to the state of a learner. The CBM network may compute registered subscriptions matching the update events and generate corresponding notifications, which are sent to application registered from the update event.

The state system 240 also may include one or more servers that provide state access functions 246. The state access functions may be called using the CBM-RPC protocol described above, for example, to access information (e.g., historical data) stored in the state database. The accessed information may be used to implement a start state within an application.

Because a CBM system is stateless, any application starting after learner events have been published to the network is not aware of the past learner events. In order to establish the current learner state and any other historical data (e.g., past learner events stored in the state database), an application may call a state access function using the CBM-RPC protocol. The state access function called by the CBM-RPC accesses the requested information from the database 248 and publishes a response to the CBM network, which notifies the requesting application. The response may be used as the initial information needed by the requesting application.

An application or state service may publish event information to the CBM network 230, and the CBM network 230 may provide notifications to any subscribing applications using the event formats described in detail below. The published information provided in the notifications may be used, for example, to provide an online tutor with data regarding the actions of learners assigned to the tutor. Notification formats may be defined for each e-learning application for any events that a subscribing application wants to monitor, according to the examples described below.

Events may be grouped into one or more levels (e.g., low-level and high-level events). Low-level events may include events that are not associated with a specific

service and that do not require additional processing (e.g., a Login event). High-level events may be associated with specific services (e.g., a state service) and need additional processing. An example of a high-level service is the courseMembersChanged service. High-level events may be created by services from low-level events (e.g., the courseMembersChanged event may be derived from a courseJoinedEvent and a courseLeftEvent).

Each event may be assigned a unique event identifier. An event identifier may include one or more parts, such as, for example, an eventnamespace, an eventfamily, and an eventtype. An eventnamespace may be used to distinguish between events from different applications that have the same name and to group events from one application (e.g., so that other software can easily recognize events associated with the application). Event-namespaces may use a uniform resource locator (URL) as the namespace identifier. Each learning environment may have its own event namespace and its own semantics (i.e., the specification of the conditions that meet the requirements for publishing an event) associated with events

Events also may be grouped into logical event families (e.g., presence events, course events, navigation events, newsgroup events, and chat events). The event family may be used for handling events. A specific service may only register for an event family (e. g., if the family is "presence", the events are "log in" and "log out"). The service may use these events to determine information (e.g., using the previous example, a service may calculate the time as user was online).

The event-type defines a class of the event and serves as a basic identifier. All events are constructed with a string reference to the "source" that generated the event.

As described above, the CBM network includes one or more different data types, such as, for example, "int32", "int64", "real64" and "string" that may be used for values in notifications associated with published events. A composite data type may be used to represent event information that may not be expressed as one of the four data types described above, such as, for example a record or an array. For example, an array may be used to describe learners taking a course, identified as a list returned by the "get*CourseMembers*" procedure call. The list may have the following composite data type:

<composite>
<string> *CourseMember1* </string>
<string> *CourseMember2* </string>

</composite>

CBM Format Specification

All events used by the CBM e-learning system may be formatted using the general event format listed below.

Name	Default	Type	Opt/Mand	Description
CBM.event	0	int32	M	major protocol version
Minor	1	int32	M	minor protocol version
Source	URI	String	M	unique ID of client instance, that generated the event. Suggested is the URI of the publishing application, e.g., the URI of a servlet-process. Applications may define their own unique identifier. This field can be used by consumers to resolve conflicting information from users that are running multiple clients.
Time		int64	M	Timestamp of the event.
eventnamespace	URI	String	M	namespace of event
Eventfamily		String	M	eventfamily of event
Eventtype		String	M	type of event this message applies to
documentation		String	O	human readable information about this event
Params		composite	M	identifiers of parameters in this message. This field is a composition of string elements and stored in an ordinary string

Any number of different events may be provided for by the CBM system. Examples of events may include presence events, course events, navigation events, newsgroup events, and chat events. Exemplary formats for each of these families of events is describe in detail below.

Presence events may be used to indicate the presence of a learner (e.g., connection to the e-learning system). For example, presence events may include a LoginEvent and a LogoutEvent. A "LoginEvent" may occur when a user (e.g., a learner or a tutor) connects to the learning system. A login event may be expressed using the following format: eventfamily = "presence" of type "string"; eventtype = "LoginEvent" of type string;

parameters including ["user"] of type composite; and having a user = "userID" of type string (e.g., any string that uniquely identifies the user, such as an E-Mail-address). The following example illustrates a login event format:

cbm.event:	1
minor:	0
source:	"http://192.168.0.0/LoginServlet"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"presence"
eventtype:	"LoginEvent"
documentation:	"User has logged in"
params:	"<composite><string>user</string></composite>"
user:	"twiki@learningsolution"

A logout event may occur when a user leaves the learning environment (e.g., disconnects with the learning system). The logout event may have the following format: eventfamily = "presence" of type "string"; eventtype = "LogoutEvent" of type string; parameters including ["user"] of type composite; and having a user = "userID" of type string (e.g., any string that uniquely identifies the user, such as an E-Mail-address). An example of a logout event may be expressed as follows:

cbm.event:	1
minor:	0
source:	"http://192.168.0.0/LoginServlet"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"presence"
eventtype:	"LogoutEvent"
documentation:	"User has logged out"
params:	"<composite><string>user</string></composite>"
user:	"twiki@learningsolution"

Course events may be used to indicate information about courses within the e-learning system and may include CourseJoinedEvent, CourseLeftEvent, and CourseMemberChangedEvent.

A "CourseJoinedEvent" occurs when a user enters a course. For learners this event may indicate that the learner is interacting with the course. For tutors this event

may indicate that a tutor is present within a course to assist learners taking the course. The CourseJoinedEvent may have the following format: eventfamily = "course" of type "string"; eventtype = "CourseJoinedEvent" of type string; parameters may include ["user", "course"] of type composite; a user = "userID" of type string (e.g., any string that uniquely identifies the user, such as an E-Mail-address), and a course = "courseID" of type string (e.g., a unique course name). An example of a CourseJoinedEvent may be expressed as follows:

cbm.event:	1
minor:	0
source:	"http://192.168.0.0/CourseLoginServlet"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"course"
eventtype:	"CourseJoinedEvent"
documentation:	"User has entered a course"
params:	"<composite><string>user</string> <string>course</string></composite>"
user:	"twiki@learningsolution"
course:	"sap20courses/beginnerscourse"

A CourseLeftEvent occurs when a user leaves a course. For learners this event may indicate that the learner has stopped interacting with the course. For tutors this event may indicate that a tutor is unavailable to assist learners of the course. The CourseLeftEvent may have the following format: eventfamily = "course" of type "string"; eventtype = "CourseLeftEvent" of type string; parameters including ["user", "course"] of type composite; a user = "userID" of type string (e.g., any string that uniquely identifies the user, such as an E-Mail-address), and a course = "courseID" of type string (e.g., a unique course name). An example of a CourseLeftEvent may be expressed as follows:

Example:

cbm.event:	1
minor:	0
source:	"http://192.168.0.0/CourseLogoutServlet"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"

eventfamily:	"course"
eventtype:	"CourseLeftEvent"
documentation:	"User has left a course"
params:	"<composite><string>user</string>
	<string>course</string></composite>"
user:	"twiki@learningsolution"
course:	"sap20courses/beginnerscourse"

The CourseMembersChangedEvent is a high-level event that may be published by a state service that subscribes to the CourseJoinedEvent and the CourseLeftEvents. The state service may update an associated course member table from these events and publish the update as a CourseMembersChangedEvent notification. Course members may be those members that are currently online actively participating in the course. The CourseMembersChangedEvent may have the following format: eventfamily = "course" of type "string"; eventtype = "CourseMembersChangedEvent" of type string; parameters including ["course", "members"] of type composite; a course = "courseID" of type string (e.g., a unique course name); and members = ["user1", "user2", ...] of type composite. An example of a CourseMembersChangedEvent may be expressed as follows:

cbm.event:	1
minor:	0
source:	"http://192.168.0.0/CourseService"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"course"
eventtype:	"CourseMembersChangedEvent"
documentation:	"List of all users in the course"
params:	"<composite><string>course</string>
	<string>members</string></composite>"
course:	"sap20courses/beginnerscourse"
members:	"<composite><string>twiki@learningsolution</string>
	<string>markus@tutorsolution</string></composite>"

Navigation Events may be used to indicate the actions of a user within a course. Navigation events may include CoursePositionChangedEvent, ExerciseStartedEvent, and ExerciseFinishedEvent.

E-learning courses may be divided into different sections that a learner progresses through as the learner takes the course. The CoursePositionChangedEvent may be used to indicate when a learner moves from one section to another. This format does not dictate the way in which a course is divided and structured. Instead, the position inside a course may be defined by a unique positioning string. The CoursePositionChangedEvent may have the following format: eventfamily = "navigation" of type "string"; eventtype = "CoursePositionChangedEvent" of type string; parameters including ["learner", "course", "position"] of type composite; a learner = userID of type string (e.g., a unique identifier, such as an email address; a course = "courseID" of type string (e.g., a unique course name); and a position = "positionID" of type string. An example of a CoursePositionChangedEvent may be expressed as follows:

CBM.event:	1
minor:	0
source:	"http://192.168.0.0/Mediator"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"navigation"
eventtype:	"CoursePositionChangedEvent"
documentation:	"Subject, learner is dealing with"
params:	"<composite><string>learner</string><string>course</string><string>position</string></composite>"
learner:	"twiki@learningsolution"
course:	"sap20courses/beginnerscourse"
position:	"sap20courses/beginnerscourse/chapter23"

E-learning courses may include various exercises (e.g., a working example, a practice problem, a quiz, or a test) that a learner performs while taking the course. The ExerciseStartedEvent may be used to indicate when a learner starts an exercise of a course. The ExerciseStartedEvent may have the following format: eventfamily = "navigation" of type "string"; eventtype = "ExerciseStartedEvent" of type string; parameters including ["learner", "course", "exercise"] of type composite; a learner =

userID of type string (e.g., a unique identifier, such as an email address; a course = "courseID" of type string (e.g., a unique course name); and a exercise = "exerciseID" of type string (e.g., a unique exercise name). An example of an ExerciseStartedEvent may be expressed as follows:

CBM.event:	1
minor:	0
source:	"http://192.168.0.0/Mediator"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"navigation"
eventtype:	"ExerciseStartedEvent"
documentation:	"User has started an exercise"
params:	"<composite><string>learner</string> <string>course</string><string>exercise </string></composite>"
learner:	"twiki@learningsolution"
course:	"sap20courses/beginnerscourse"
exercise:	"sap20courses/beginnerscourse/chapter23/exercise2"

An ExerciseFinishedEvent may be used to indicate when a learner has completed or finished an exercise of a course. The ExerciseFinishedEvent may have the following format: eventfamily = "navigation" of type "string"; eventtype = "ExerciseFinishedEvent" of type string; parameters including ["learner", "course", "exercise"] of type composite; a learner = userID of type string (e.g., a unique identifier, such as an email address; a course = "courseID" of type string (e.g., a unique course name); and a exercise = "exerciseID" of type string (e.g., a unique exercise name). An example of an ExerciseStartedEvent may be expressed as follows:

CBM.event:	1
minor:	0
source:	"http://192.168.0.0/Mediator"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"navigation"
eventtype:	"ExerciseFinishedEvent"
documentation:	"User has finished an exercise"
params:	"<composite><string>learner</string>

	<string>course</string><string>exercise
	</string></composite>"
learner:	"twiki@learningsolution"
course:	"sap20courses/beginnerscourse"
exercise:	"sap20courses/beginnerscourse/chapter23/exercise2

Newsgroup events may indicate when a user interacts with a newsgroup (e.g., articles from a publication). Newsgroup events may include a NewsgroupOpenedEvent, a NewsgroupArticleOpenedEvent, and a NewsgroupArticlePostedEvent.

Newsgroups may be used to exchange information within an e-learning environment. For example, newsgroups may be implemented using a blackboard/message board where learners and tutors may read and post articles (e.g., notes, questions, answers, and comments) regarding a course. Tutors may be interested whether a learner has opened a newsgroup and has read article headlines associated with a newsgroup. The NewsgroupOpenedEvent may be used to indicate when a user opens a newsgroup and reads the article names. The NewsgroupOpenedEvent may have the following format: eventfamily = "newsgroup" of type "string"; eventtype = "NewsgroupOpenedEvent" of type string; parameters including ["user", "newsgroup"] of type composite; a user = userID of type string (e.g., a unique identifier, such as an email address; and a newsgroup = "newsgroupID" of type string (e.g., a unique newsgroup name). An example of an NewsgroupOpenedEvent may be expressed as follows:

CBM.event:	1
minor:	0
source:	"http://192.168.0.0/Mediator"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"newsgroup"
eventtype:	"NewsgroupOpenedEvent"
documentation:	"User has opened a newsgroup"
params:	"<composite><string>user</string>
	<string>newsgroup</string></composite>"
user:	"twiki@learningsolution"
newsgroup:	"sap20courses/beginnerscourse/newsgroup"

The NewsgroupArticleOpenedEvent may be used to indicate when a user opens an article in a newsgroup in order to read the article. The NewsgroupOpenedEvent may

have the following format: eventfamily = "newsgroup" of type "string"; eventtype = "NewsgroupArticleOpenedEvent" of type string; parameters including ["user", "newsgroup", and "articlename"] of type composite; a user = userID of type string (e.g., a unique identifier, such as an email address; a newsgroup = "newsgroupID" of type string (e.g., a unique newsgroup name); and an articlename = "articleID" of type string (e.g., a unique article name, such as an article headline). An example of an NewsgroupArticleOpenedEvent may be expressed as follows:

CBM.event:	1
minor:	0
source:	"http://192.168.0.0/NewsgroupManagerServlet"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"newsgroup"
eventtype:	"NewsgroupArticleOpenedEvent"
documentation:	"User has opened a newsgroup article"
params:	"<composite><string>user</string> <string>newsgroup</string><string>articlename </string></composite>"
user:	"twiki@learningsolution"
newsgroup:	"sap20courses/beginnerscourse/newsgroup"
articlename:	"welcome to this newsgroup"

The NewsgroupArticlePostedEvent may be used to indicate when a user sends an article to a newsgroup. The NewsgroupArticlePostedEvent may have the following format: eventfamily = "newsgroup" of type "string"; eventtype = "NewsgroupArticlePostedEvent" of type string; parameters including ["user", "newsgroup", "articlename", and "articletext"] of type composite; a user = userID of type string (e.g., a unique identifier, such as an email address; a newsgroup = "newsgroupID" of type string (e.g., a unique newsgroup name); an articlename = "articleID" of type string (e.g., a unique article name, such as an article headline), and article text = "text of article" of type string. An example of an NewsgroupArticlePostedEvent may be expressed as follows:

CBM.event:	1
minor:	0
source:	"http://192.168.0.0/NewsgroupManagerServlet"

time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"newsgroup"
eventtype:	"NewsgroupArticlePostedEvent"
documentation:	"User has posted a newsgroup article"
params:	"<composite><string>user</string><string>newsgroup</string><string>article</string><string>articletext</string></composite>"
user:	"twiki@learningsolution"
newsgroup:	"sap20courses/beginnerscourse/newsgroup"
articlename:	"welcome to this newsgroup"
articletext:	"I want to welcome you to ..."

The e-learning system 200 may provide for interactive events between users (e.g., Chat Events). Chat events may include ChatJoinedEvent, ChatLeftEvent, ChatMessageEvent, and ChatMembersChangedEvent.

The ChatJoinedEvent may be used to indicate when a user joins a chat session. The ChatJoinedEvent may have the following format: eventfamily = "chat" of type "string"; eventtype = "ChatJoinedEvent" of type string; parameters including ["user", "chat"] of type composite; a user = userID of type string (e.g., a unique identifier, such as an email address; and a chat = "chatID" of type string (e.g., a unique chat identifier).. An example of an ChatJoinedEvent may be expressed as follows:

CBM.event:	1
minor:	0
source:	"http://192.168.0.0/Mediator"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"chat"
eventtype:	"ChatJoinedEvent"
documentation:	"User has joined a chat"
params:	"<composite><string>user</string><string>chat</string></composite>"
user:	"twiki@learningsolution"
chat:	"sap20courses/beginnerscourse/chatgroup"

The ChatLeftEvent may be used to indicate when a user exits a chat session. The ChatLeftEvent may have the following format: eventfamily = "chat" of type "string"; eventtype = "ChatLeftEvent" of type string; parameters including ["user", "chat"] of type composite; a user = userID of type string (e.g., a unique identifier, such as an email address; and a chat = "chatID" of type string (e.g., a unique chat identifier). An example of a ChatLeftEvent may be expressed as follows:

CBM.event:	1
minor:	0
source:	"http://192.168.0.0/Mediator"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"chat"
eventtype:	"ChatLeftEvent"
documentation:	"User has left a chat"
params:	"<composite><string>user</string>

The ChatMessageEvent may be used to indicate when a user sends a message in a chat room. The ChatMessageEvent may have the following format: eventfamily = "chat" of type "string"; eventtype = "ChatMessageEvent" of type string; parameters including ["user", "chat", "Chatmessage"] of type composite; a user = userID of type string (e.g., a unique identifier, such as an email address; a chat = "chatID" of type string (e.g., a unique chat identifier), and chatmessage = message text of type string. An example of a ChatMessageEvent may be expressed as follows:

CBM.event:	1
minor:	0
source:	"http://192.168.0.0/ChatApplet"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"chat"
eventtype:	"ChatMessageEvent"
documentation:	"User has sent a chat message"
params:	"<composite><string>user</string> <string>chat</string><string>chatmessage </string></composite>"
user:	"twiki@learningsolution"
chat:	"sap20courses/beginnerscourse/chatgroup"

chatmessage: "Hi there ... :-)"

The ChatMembersChangedEvent is a high-level event that is usually published by a state service that subscribes to a ChatJoinedEvent and a ChatLeftEvent. The ChatMembersChangedEvent state service updates a chat room member table from the ChatJoinedEvent or the ChatLeftEvent and publishes a ChatMembersChangedEvent notification. The ChatMembersChangedEvent may have the following format: eventfamily = "chat" of type "string"; eventtype = "ChatMembersChangedEvent" of type string; parameters including ["chat", "members"] of type composite; a chat = "chatID" of type string (e.g., a unique chat identifier), and members = ["user1", "user2", ...] of type composite. An example of a ChatMembersChangedEvent may be expressed as follows:

CBM.event:	1
minor:	0
source:	"http://192.168.0.0/CourseService"
time:	157235472547
eventnamespace:	"http://www.sap.com/cbm/elearning"
eventfamily:	"chat"
eventtype:	"ChatMembersChangedEvent"
documentation:	"List of all users in the chatgroup"
params:	"<composite><string>chat</string><string>members</string></composite>"
chat:	"sap20courses/chat"
members:	"<composite><string>twiki@learningsolution</string><string>markus@tutorsolution</string></composite>"

Tutor Services

An important aspect of providing a suitable e-learning environment includes providing adequate support for the learners using the system. One aspect of proving adequate support is to oversee the progress of a learner and provide guidance and assistance based on the actions of a learner. However, online tutors are not able to have the personal or face-to-face contact with the learners to provide the oversight typically available from classroom based training.

Support for online learning is important to ensure that the needs of each learner are addressed. Although each learner may take the same training and/or courses, each learner has individual preferences, strengths, weaknesses, abilities, skills, knowledge, and

know-how. As a result, different learners have different needs and require different support for their online training.

To provide for individual learner needs and generally improve the support for learning in an e-learning environment, the progress and interaction of learners with the learning material and learning system may be monitored using a CBM system 200. In particular, the actions of online learners and an overview of their dynamic learning state may be monitored to provide support for the individual learner. Tutors may observe the actions of online learners and their dynamic learning state to provide support and guidance when needed using the CBM system 200.

Monitoring of individual learners and their interaction with the learning system may be provided by a number of services provided, for example, by the state services of the state system 240. Examples of state services that may be provided to monitor learner events include a WatchlistService, a CourseMemberService, a DiaryService, and a LearnerService.

The WatchlistService generates a watchlist for a tutor. The watch list provides a list of all courses that the tutor wishes to monitor. The Watchlist may be implemented using two CBM-RPCs, for example, "getWatchlist" and "setWatchlist." The procedure getWatchlist may include the parameter tutor of type string and generate a return Watchlist of type composite. The procedure setWatchlist may include parameters tutor of type string and watchlist of type composite and return an indication Ok of type int32. Course names returned by the watchlist CBM-RPC may be stored in a file "watchlist.txt." The file may be stored in a directory of the server where the watchlist service is running. The file may be used to initialize a tutor application with the course names to populate in a watchlist window.

The CourseMemberService may be used to determine the members of a course, such as, for example, the names of the learners that are registered to take the course. The CourseMembers may be determined using four CBM-RPCs, for example, "getLearnersRegistered, " "setLearnersRegistered," "getLearnersOnline," and "setLearnersOnline." The procedure getLearnersRegistered may include the parameter coursename of type string and generate a return of learner names of type composite. The procedure setLearnersRegistered may include parameters coursename of type string and learnerlist of type composite and return an indication Ok of type int32. The course names returned by the CBM-RPC to the CourseMemberService may be stored in a file "learnerlist.txt." The file may be stored in a directory of the server where the

CourseMemberService is running. The file may be used to initialize a tutor application to populate a course window with the names of learners of a course assigned to a tutor.

The CourseMemberService also may be used to determine the learners as a subgroup of the registered learners that are currently online (e.g., learners actively participating in the course) using the procedures getLearnersOnline and setLearnersOnline. The procedure getLearnersOnline may include the parameter coursename of type string and generate a return learners of type composite. The procedure setLearnersOnline may include parameters coursename of type string and learnerlist of type composite and return an indication Ok of type int32. Course names returned by a CBM-RPC of the CourseMemberService may be stored in a file "coursejoined.txt." The file may be stored in a directory of the server where the CourseMemberService is running. The file may be used to initialize a tutor application with the learners' names to implement a course window.

The course member service also may publish update events to indicate when the state of the group of online learners changes. For example, the service may publish an update event to indicate that a learner's state has changed from online to offline. The tutor application may subscribe to the update event and use a notification to change the indication of the learner's current status in the course window.

The Diary service stores a summary of the actions for each learner of the e-learning system for a specified period of time (e.g., a day, a week, and/or a month). The summary may include the name and the time of each action with the course performed by a learner. The Diary may be implemented using two CBM-RPCs getSummary and clearDiary. The procedure getSummary may include the parameters learner of type string, course of type string, and timemillis of type int64. The procedure clearDiary may include parameter learner of type string and return an indication Ok of type int32. The diary returned by a CBM-RPC of the DiaryService may be stored in a file "learnerdiary.txt." The file may be stored in a directory of the server where the Diary service is running. The file may be used to initialize a tutor application with the past acts of a learner over time to implement a diary window.

The LearnerService may be used to store the actual current occupation for each learner (e.g., the position of a learner or the current interaction of a learner with a course). The LearnerOccupation may be implemented using two CBM-RPCs of getCurrentOccupation and setCurrentOccupation. The method getCurrentOccupation may include the parameter learner of type string and generate a return of occupation of

type string. The method setCurrentOccupation may include parameters learner of type string and occupation of type string and return an indication Ok of type int32. Occupations returned by a CBM-RPC of the LearnerService may be stored in a file "occupations.txt." The file may be stored in a directory of the server where the Diary service is running. The file may be used to initialize a tutor application to populate a course window with the occupation of each learner in a course.

Tutor Monitoring Process

Fig. 3 shows a process 300 that may be used by a tutor to monitor an e-learning system. The tutor using a client device (e.g., a computer or work station) may initiate a tutor application by selecting the application and starting the application (301). The tutor application may publish a message to the CBM system 200 that the tutor is online and available to provide assistance. The tutor application subscribes to the CBM network and a number of services (e.g., a WatchlistService, a CourseMemberService, a DiaryService, and a LearnerService) provided by a state system 240 (310). The tutor application may make one or more CBM-RPCs of state access functions of the state system 240 using the CBM-RPC protocol (320). The tutor application receives data from the state system 240 returned as responses to the CBM-RPCs (330). The data may be used to populate a tutor interface with information used by a tutor to monitor and manage learners. Events and update events published to the CBM network from learner applications and states services, respectively, are continually provided to the tutor application as notifications (340). The notifications are used to update information displayed in the tutor interface (350). The tutor application may determines when the tutor logs-off (360). The application may publish a notification to the system that the tutor has logged off to inform learner applications that the tutor in no longer available to provide assistance (370) and end (380).

Tutor Interface

The e-learning system may be provided with a tutor application that assists a tutor to monitor and manage one or more learners. The tutor application may provide a graphic user interface that allows the tutor to monitor and determine information about learners that use the e-learning system. The tutor interface may be implemented using a tutor application that functions in conjunction with a browser to display learner information in the interface. The tutor application may be provided with an application program

interface (API) that uses published update events from the state services described above (e.g., a WatchlistService, a CourseMemberService, a DiaryService, and a LearnerService) and CBM-RPCs to gather information that is used to populate one or more windows and/or fields in the interface.

As shown in Fig. 4, a tutor interface 400 may be provided to offer tutors an overview of the courses and activities of learners that a tutor monitors. Using the interface 400, the tutor may gain insight into the activities of learners, provide better assistance to the learners in their online activities, and generally provide support to the learners. The tutor interface 400 may provide a number of menus bars and tool bars 401 to activate and control functions of the interface 400. The tutor interface 400 also may include one or more fields and windows that provide functions and information to the tutor. For example, the tutor interface 400 also may include a messaging window 410, a course search field 420, a watch list window 430, and a course window 440. The windows may be populated with information provided by the tutor application. The information may include data derived from published learner events, published update events, and CBM-RPCs.

The messaging window may provide the tutor with an instant messaging feature. The messaging window may provide a list of users that the tutor frequently contacts or monitors online (e.g., course instructors, other tutors, and learners). Selecting a name from the list may cause a message window to appear in which the tutor may send a message or join a chat. The messaging window provides one way in which a tutor may directly contact a learner who is online and provide assistance to the learner. Scroll bars may be provided to view of any information that may not be immediately displayed within the window (e.g., due to size or the amount of information included in the window).

A search field may be used to implement a course search function. A course may be selected (e.g., from a drop-down menu) or entered in the field (e.g., using a user input device). Activation of a search button 450 causes the tutor application to search for a specified course or provide a list of courses meeting a search criterion. The tutor may select a course that is returned by the search function and add the course to the watch list.

The watch list window may include a list of all courses (e.g., all course for which the tutor is responsible). A watch list may be maintained for each tutor. A course may be selected from the list using an input device (e.g., a mouse, a touchpad, a keypad, a

pointer, or a keyboard). Selecting a course from the watch list populates the course window within information regarding the selected course.

A course window may provide the tutor with information about learner activities for a selected course. The window may include a name or identification of the course for which the window is populated. The window may display a list 460 of one or more indications (e.g., an ID or a name of a learner) associated with all learners enrolled in a course. Selection of the name from the list may activate a messaging feature (e.g., e-mail, chat, or instant message) allowing the tutor establish a dialog with the selected learner. The display also may include an indication 462 (e.g., a check or a minus) of whether the learner is currently engaged in any activities with the course. If the learner is engaged in a course activity, the display also may include a description of the activity 464 (e.g., an indication of the learner's position within a course, such as "Chapter 3").

The course window also may include one or more buttons or other input features that allow the tutor to interact with the course window. For example, a course add 470 and course remove 475 button may be provided allowing the tutor to add or remove the course from the watch list, respectively.

A course diary icon 480 may be provided for each learner. Selection of the icon 480 causes a diary window 500 to appear, as shown in Fig. 5. The diary window 500 may include a number of sections 510 that describe the historical activities of a learner within the course. For example, the activities of a learner may be divided into various periods of time (e.g., days, weeks, and/or months). The period of time may be selected from a menu or set a property of the tutor application. The activity of a learner at a particular date and time may be provided within one the sections 510 corresponding to the time period. In this way, the tutor may be provided with a general overview of the activities of learners within a course over the period of time.

The e-learning CBM-RPC protocol, tutor application, and tutor interface provides an architecture that supplies a tutor with valuable insight to the activities of the learners of an e-learning system not available from other computer based training systems. The insight allows the tutor to monitor the dynamic state of a leaner and may be used to provide individualize support for the learners, enhance their e-learning experience, and to augment the knowledge gained from using an e-learning system.

A number of implementations have been described. Nevertheless, it will be understood that various modifications may be made. For example, suitable results may be achieved if the steps of the disclosed techniques are performed in a different order

and/or if components in a disclosed system, architecture, device, or circuit are combined in a different manner and/or replaced or supplemented by other components. Accordingly, other implementations are within the scope of the following claims.

Claims

WHAT IS CLAIMED IS:

1. A tutor system comprising:
an input to receive one or more notifications of published learner events; and
a processor to generate subscriptions to the notifications, to process the received notifications, and to generate a tutor interface based on the processed notification, the interface including a course area to display dynamic learner state information.
2. The system of claim 1 wherein the course area displays one or more learners monitored by the tutor.
3. The system of one of the preceding claims wherein the course area displays an indication of whether a learner is online.
4. The system of one of the preceding claims wherein the course area displays an indication of whether a learner is interacting with a course.
5. The system of one of the preceding claims wherein the course area displays a current learner action with a course.
6. The system of one of the preceding claims wherein the course area displays a position of a learner within a course.
7. The system of one of the preceding claims wherein the course area displays a learner diary feature.
8. The system of claim 7 wherein selection of the learner diary feature causes the processor to generate a learner diary display.
9. The system of claim 8 wherein the learner diary display displays one or more actions of a learner over a period of time.
10. The system of claim 9 wherein the period of time is a week and each day is displayed as a section in the diary display.

11. The system of one of the preceding claims wherein the processor generates a watch list display including one or more courses monitored by the interface.
12. The system of claim 11 wherein selection of a course from the watch list causes the processor to populate the course area with information based on the selected course.
13. The system of one of the preceding claims wherein the processor generates a message area to provide a message service.
14. The system of one of the preceding claims wherein the processor is configured to generate a CBM-remote procedure call (RPC) to access a remote function.
15. The system of claim 14 wherein the input is configured to receive a return notification based on the CBM-RPC and the processor is configured to populated the interface with information based on the return notification.
16. The system of claim 15 wherein the information based on the return notification is learner state information.
17. The system of one of the preceding claims wherein the input is configured to receive notifications based on learner update events produced by a remote service and the processor is configured to populate the interface with information based on the learner update event notifications.
18. The system of claim 17 wherein the learner update information is based on a modification to a dynamic state of one or more learners.
19. A graphical user interface for a tutor application comprising:
a course area to display information about a course based on a notifications of published events received from a content based messaging network, the course area displaying an identification of one or more learners taking the course and information about the one or more learners on the list.
20. The graphical user interface of claim 19 wherein the course area displays a list of learners monitored by the tutor.
21. The graphical user interface of one of claims 19, 20 wherein the course area displays an indication of whether a learner is online.
22. The graphical user interface of one of claims 19 to 21 wherein the course area displays an indication of whether a learner is interacting with a course.
23. The graphical user interface of one of claims 19 to 22 wherein the course area displays a current learner action with a course.

24. The graphical user interface of one of claims 19 to 23 wherein the course area displays a position of a learner within a course.
25. The graphical user interface of one of claims 19 to 24 wherein the course area displays a learner diary icon.
26. The graphical user interface of claim 25 wherein selection of the learner diary icon causes the display of a learner diary window.
27. The graphical user interface of claim 26 wherein the learner diary window displays one or more actions of a learner over a period of time.
28. The graphical user interface of claim 27 wherein the period of time is a week and each day is displayed as a section in the diary window.
29. The graphical user interface of one of claims 19 to 28 further comprises a watch list display to display one or more courses monitored by the interface.
30. The graphical user interface of claim 29 wherein selection of a course from the watch list causes the course area to populate with information based on the selected course.
31. The graphical user interface of one of claims 19 to 30 further comprising a message window to provide a message service.
32. The graphical user interface of one of claims 19 to 31 further comprising a search field to search for a course.
33. The graphical user interface of claim 32 further comprising a search input icon to activate a search for a course in the search field.
34. The graphical user interface of claim 33 wherein one or more courses returned in response to selection of the search icon may added to a watch list of courses monitored by the interface.
35. The graphical user interface of one of claims 19 to 34 wherein the interface is configured to populate course area with information based on the learner update event notifications received from a content based messaging system.
36. The graphical user interface of claim 35 wherein the learner update information is based on a modification to a dynamic state of one or more learners.
37. A method of providing a tutoring interface comprising:
subscribing to one or more learner events;
receiving one or more notifications based on the subscribed to events; and
generating a tutor interface based on received notifications, the interface including learner information.

38. The method of claim 37 wherein subscribing to one or more learner events includes subscribing to a content based message network.

39. The method of one of claims 37, 38 wherein generating a tutor interface includes generating a course area.

40. The method of claim 39 wherein generating the course area displays a list of learners monitored by the tutor.

41. The method of one of claims 39, 40 wherein generating the course area displays an indication of whether a learner is online.

42. The method of one of claims 39 to 41 wherein generating the course area displays an indication of whether a learner is interacting with a course.

43. The method of one of claims 39 to 42 wherein generating the course area displays a current learner action with a course.

44. The method of one of claims 39 to 43 wherein generating the course area displays a position of a learner within a course.

45. The method of one of claims 39 to 44 wherein generating the course area displays a learner diary icon.

46. The method of claim 45 further comprising selecting the learner diary icon and displaying a learner diary window.

47. The method of claim 46 wherein displaying the learner diary window includes displaying one or more actions of a learner over a period of time.

48. The method of claim 47 wherein displaying one or more actions of a learner over a period of time includes displaying a week and displaying each day of the week as a section in the diary window.

49. The method of one of claims 37 to 48 further comprising generating a watch list display to display one or more courses monitored by the interface.

50. The method of claim 49 further comprising selecting a course from the watch list and populating the course area with information based on the selected course.

51. The method of one of claims 37 to 50 further comprising generating a message window to provide a message service.

52. The method of one of claims 37 to 51 further comprising receiving learner update event notifications from a content based messaging system and populating the course area with information derived from the received update events.

53. The method of claim 52 wherein the receiving update events includes receiving an update event from a state service.

54. The method of one of claims 37 to 53 further comprising making a content based message remote procedure call to a CBM network.

55. The method of claim 54 further comprising receiving a return notification based on the remote procedure call and populating the interface with information based on the return notification.

56. The method of claim 55 wherein populating the interface with information based on the return notification includes populating the interface with learner state information.

57. The method of one of claims 37 to 56 further comprising publishing a login event to a content based message network to indicate a tutor available to give assistance to learners.

58. The method of one of claims 37 to 57 further comprising publishing a logout event to a content based message network to indicate a tutor is offline and unavailable to give assistance to learners.

59. A computer readable medium including instructions for causing a processor to:

receive one or more notifications of published learner events; and

generate subscriptions to the notifications, process the received notifications, and to generate a tutor interface based on the processed notification, the interface including a course area to display dynamic learner state information.

60. The computer readable medium of claim 59 further comprising instructions to cause the processor to populate the course area with a list of learners monitored by the tutor.

61. The computer readable medium of one of claims 59, 60 further comprising instructions to cause the processor to populate the course area with an indication of whether a learner is online.

62. The computer readable medium of one of claims 59 to 61 further comprising instructions to cause the processor to populate the course area with an indication of whether a learner is interacting with a course.

63. The computer readable medium of one of claims 59 to 62 further comprising instructions to cause the processor to populate the course area with a cur-

rent learner action with a course.

64. The computer readable medium of one of claims 59 to 63 further comprising instructions to cause the processor to populate the course area with the position of a learner within a course.

65. The computer readable medium of one of claims 59 to 64 further comprising instructions to cause the processor to populate the course area with a learner diary feature.

66. The computer readable medium of claim 65 further comprising instructions to cause the processor to generate a learner diary display in response to selection of the learner diary feature.

67. The computer readable medium of claim 66 further comprising instructions to cause the processor to display one or more actions of a learner over a period of time.

68. The computer readable medium of one of claim 66, 67 further comprising instructions to cause the processor to display a week and to display each day as a section in the diary display.

69. The computer readable medium of one of claims 59 to 68 further comprising instruction to cause the processor to generate a watch list display including one or more courses monitored by the interface.

70. The computer readable medium of claim 69 further comprising instructions to cause the processor to populate the course area with information based on a course selected from the watch list.

71. The computer readable medium of one of claims 59 to 70 further comprising instructions to cause the processor to generate a message area to provide a message service.

72. The computer readable medium of one of claim 59 to 71 further comprising instructions to cause the processor to generate a CBM-remote procedure call (RPC) to access a remote function.

73. The computer readable medium of claim 72 further comprising instructions to cause the processor to receive a return notification based on the remote procedure call and to populated the interface with information based on the return notification.

74. The computer readable medium of one of claims 59 to 73 wherein the instructions cause the processor to receive notifications based on learner up-

date events produced by a remote service and to populate the interface with information based on the learner update event notifications.

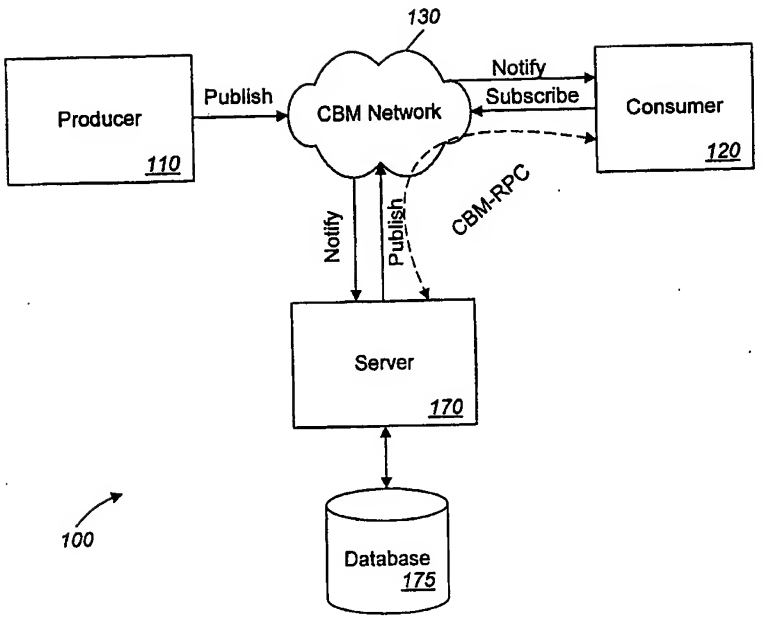

130
Producer
110
Publish
CBM Network
Notify
Subscribe
Consumer
120
Notify
Publish
CBM-RPC
Server
170
Database
175
100

FIG. 1

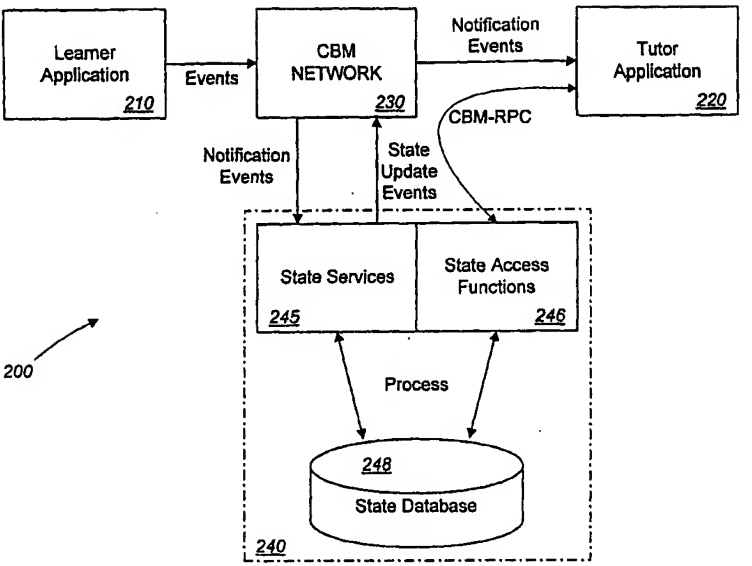

Learner Application
210
Events
CBM NETWORK
230
Notification Events
Tutor Application
220
CBM-RPC
Notification Events
State Update Events
State Services
245
State Access Functions
246
200
Process
248
State Database
240

FIG. 2

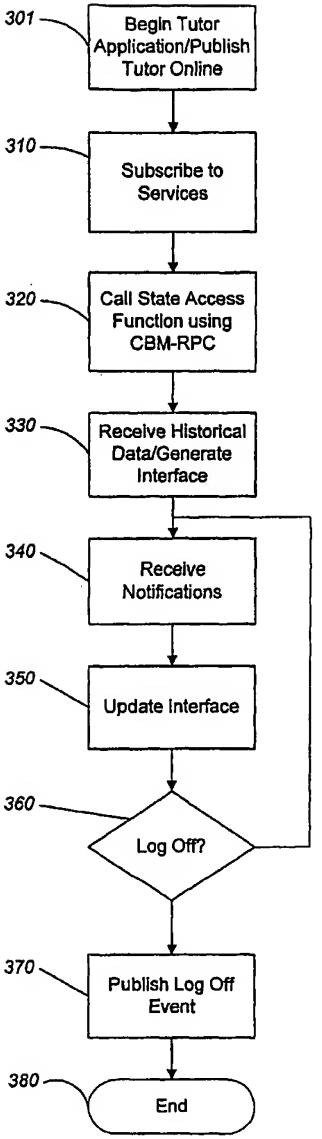

301
Begin Tutor Application/Publish Tutor Online
310
Subscribe to Services
300
320
Call State Access Function using CBM-RPC
330
Receive Historical Data/Generate Interface
340
Receive Notifications
350
Update Interface
360
Log Off?
370
Publish Log Off Event
380
End

FIG. 3

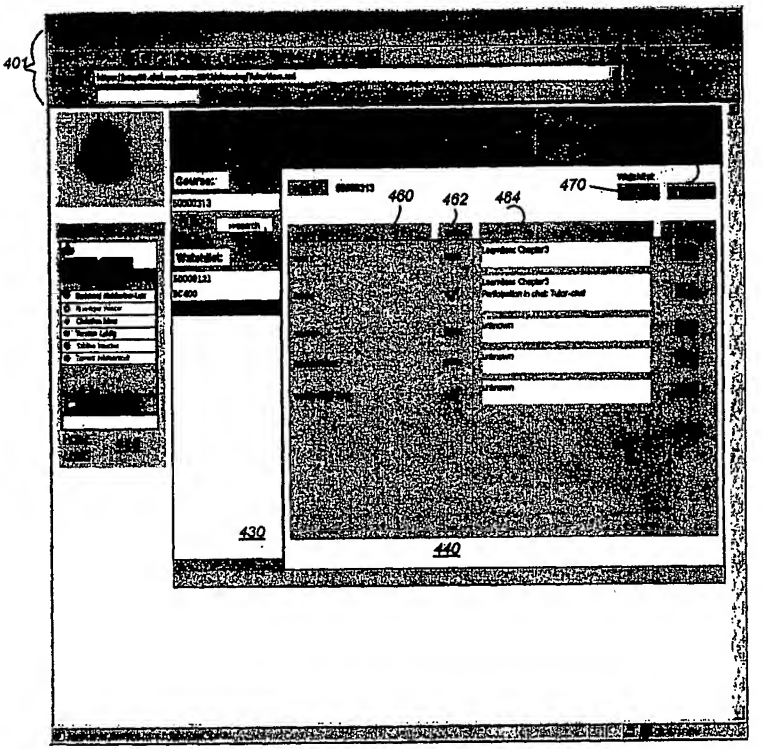


FIG. 4

400

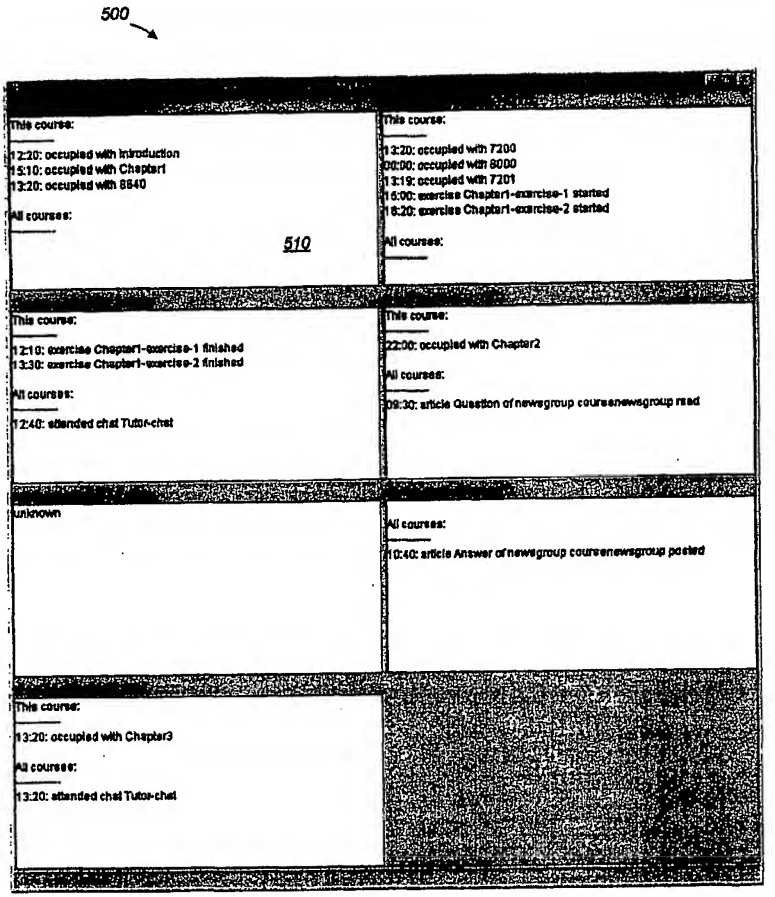

500
This course:
12:20: occupied with Introduction
15:10: occupied with Chapter1
13:20: occupied with 8640
All courses:
510
This course:
13:20: occupied with 7200
00:00: occupied with 8000
13:19: occupied with 7201
16:00: exercise Chapter1-exercise-1 started
16:20: exercise Chapter1-exercise-2 started
All courses:
This course:
12:10: exercise Chapter1-exercise-1 finished
13:30: exercise Chapter1-exercise-2 finished
All courses:
12:40: attended chat Tutor-chat
This course:
22:00: occupied with Chapter2
All courses:
09:30: article Question of newsgroup coursenewsgroup read
unknown
All courses:
10:40: article Answer of newsgroup coursenewsgroup posted
This course:
13:20: occupied with Chapter3
All courses:
13:20: attended chat Tutor-chat

FIG. 5